ROBOT SENSING AND NAVIGATION – LAB4

GPS AND IMU SENSOR FUSION

COURSE CODE: EECE5554 **NAME:** Gautham Ramkumar
PROFESSOR: Dr. Hanumant Singh **NUID:** 002308559

AIM:

To analyze the data collected using VN-100 IMU sesnor and BU-353N GPS sensor while going in a car.

DATA COLLECTION:

Two sets of data was collected:

- The first one was collected by using one of our teammate's car (Jonathan Bear) by going in a circular path near the Centennial Common of the Northeastern University. We used Jonathan's (**Gitlab ID: Bear.jo)** driver code to record both IMU and GPS data. The GPS sensor was placed on top of the car and the IMU sensor was strapped on the driver's armrest. We went around the circular pathway about 4 times.
- The second data set was collected as we drove through the city of Boston. We took many turns, and followed traffic rules during the data collection process. We also made sure that the start and end points are same.

DATA ANALYSIS:

The analysis of the collected data has been performed entirely in Python.

Q1. How did you calibrate the magnetometer data from the data collected? What were the sources of distortion present and how did you know?

We calibrated the data which we collected at Centennial Common. The sources of distortion present were Hard-Iron and Soft-Iron Errors.

Hard-Iron Errors: Hard-Iron errors are caused by objects that create a magnetic field, such as magnets or speakers. Hard iron errors cause a permanent bias in the local magnetic field, shifting the center of a circle away from the origin.

Soft-Iron Errors: Soft-Iron errors are caused by metals like iron or nickel, such as batteries. Soft iron errors distort and stretch the local magnetic field, warping the

existing magnetic fields. When plotting magnetic output, soft iron errors cause a circular output to distort into an elliptical shape.

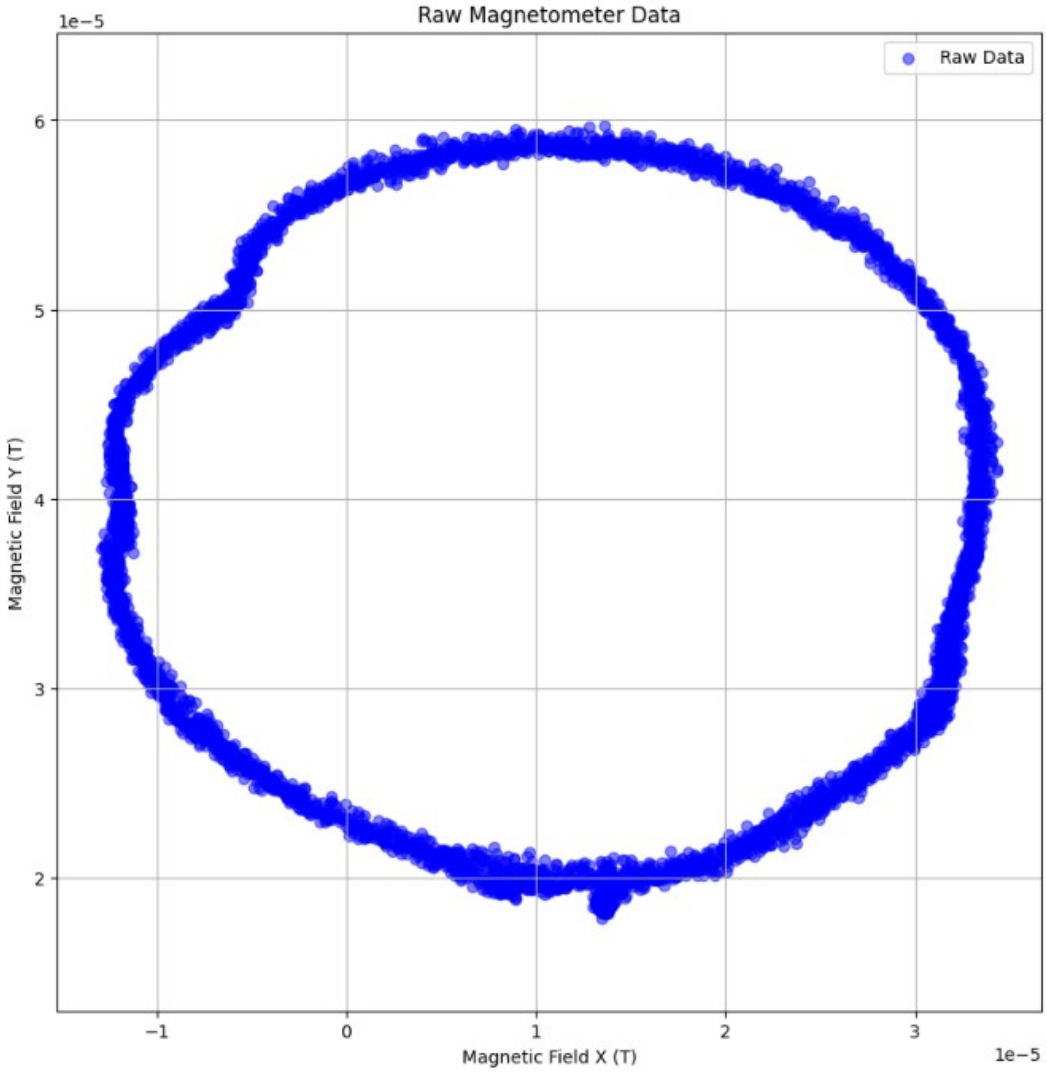

Fig 1 – Raw Magnetometer Data

Hard Iron errors can be removed by removing the offset from the raw magnetometer data.

$X_{offset} = [max(x)+min(x)]/2$; $Y_{offset} = [max(y)+min(y)]/2$

These offsets are calculated and then subtracted from the raw magnetometer data to get the hard-iron corrected data.

To get a completely corrected data, we need to solve both hard-iron and soft-iron errors. Soft iron errors change the shape of the magnetometer data. This is because they affect the orientation values of the sensor. To eliminate this error, we need to find the primary axis of the ellipse, then we need to find the distance r which we

can find by:

$$r=\sqrt{(x2-x1)^2+(y2-y1)^2}$$

After eliminating the soft-iron error, its just scaling multiplying the result with a scaling factor which is the ratio of length of major to minor axis.

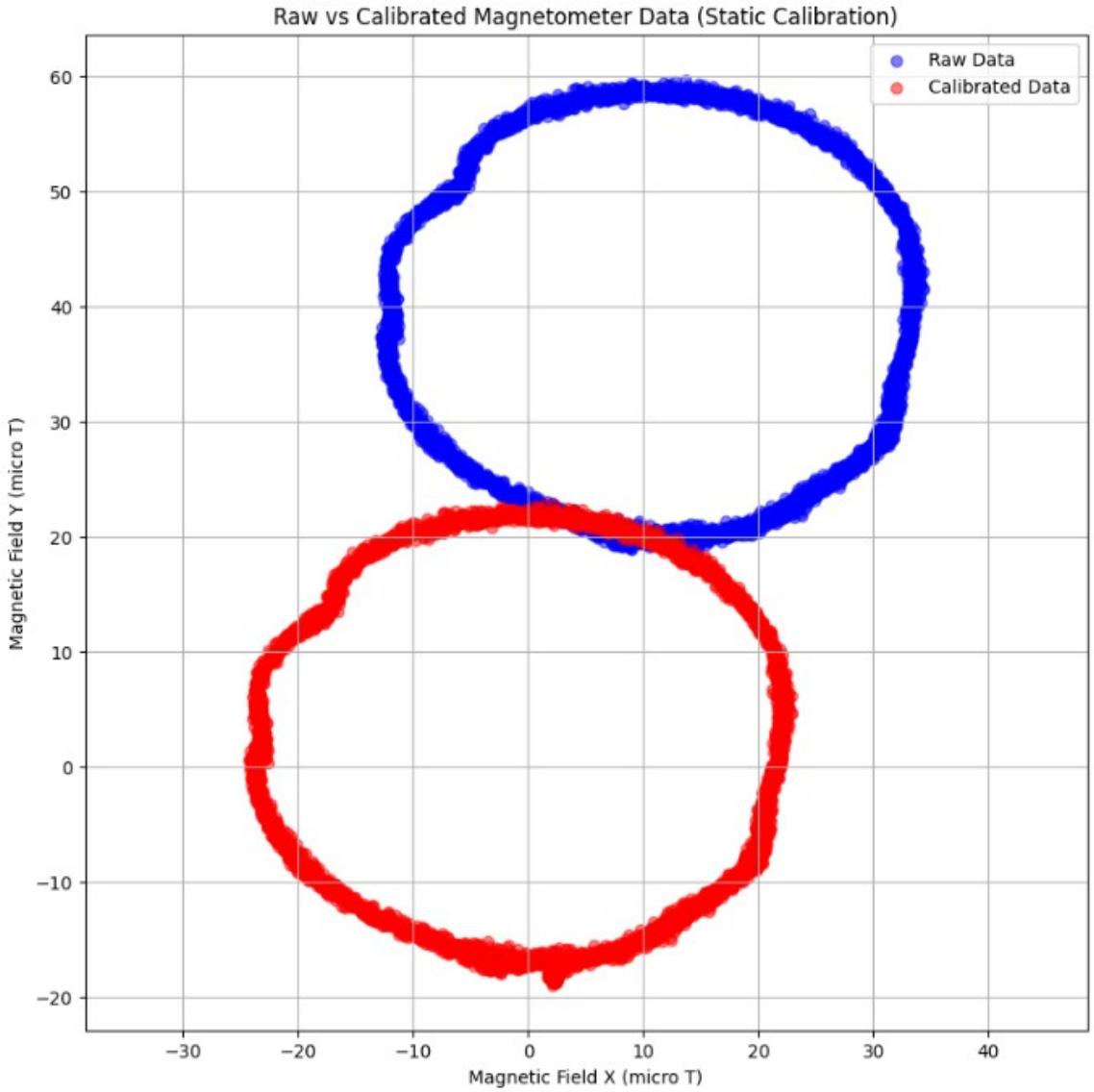

Fig – 2 Calibrated Magnetometer Data

EXTRA:

While I was looking for calibration techniques, I got to know of two methods – Static Calibration Method, Extended Kalman Filter Calibration Method.

Fig 2 shows the result from static calibration method. Static calibration is usually deployed when the sensor is kept stationary with respect to the frame. EKF calibration is an advanced calibration technique which adapts the filtering

technique according to the sensor's state (Moving/Stationary). While Static calibration technique is recommended to remove hard-iron and soft-iron errors, EKF is widely considered for many more errors.

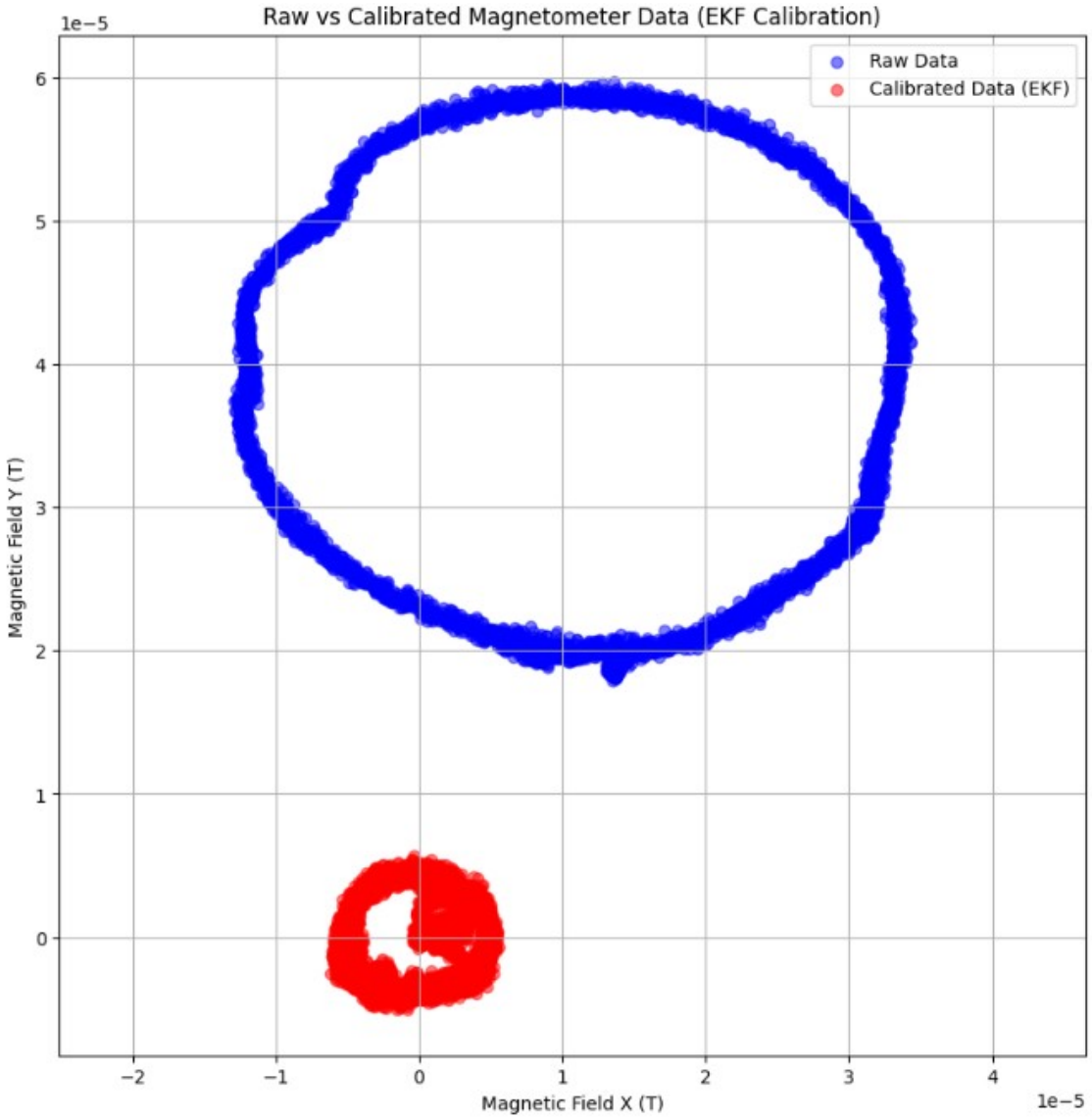

Fig – 3 EKF Calibrated Magnetometer Data

From Fig 3, it can be seen that EKF removes not only hard-iron and soft-iron errors but also other kinds of errors which we need to explore.

Q2. How did you use a complementary filter to develop a combined estimate of yaw? What components of the filter were present, and what cutoff frequencies did you use?

First I calculated the raw yaw and calibrated yaw from the magnetic field x and y values present in the IMU's Magnetometer.

Raw yaw = $\frac{magY}{magX}$

Calibrated yaw = $\frac{magY_{corrected}}{magX_{corrected}}$

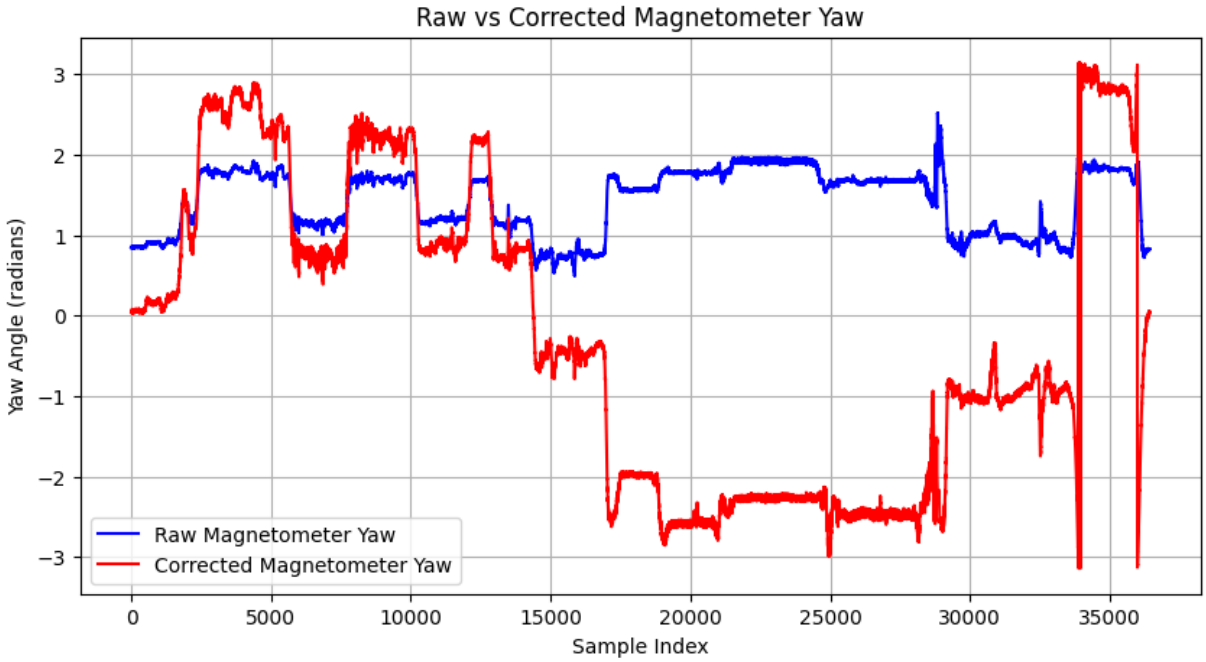

Fig – 4 Raw vs Corrected Yaw

Now this corrected yaw is unwrapped so that we can integrate it based on gyroscope's yaw angle.

Yaw angle = $\int ¿ angular\ velocity(z).d(time\ vector\ imu) > ¿$

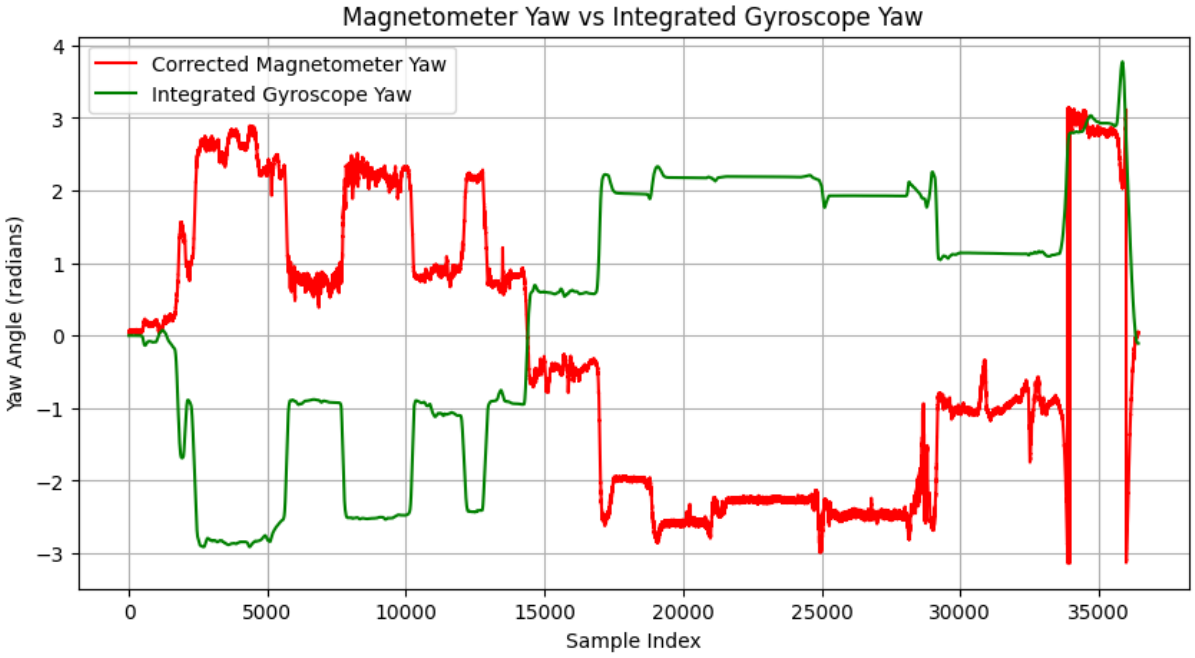

Fig – 5 Magnetometer Yaw vs Integrated Gyro Yaw

This yaw angle is then passed through a low pass filter with an alpha (α)value of 0.98 which means more weight to given gyroscope's yaw. The output graph can be seen in Fig 5. It can be seen that complementary filter produces the best results.

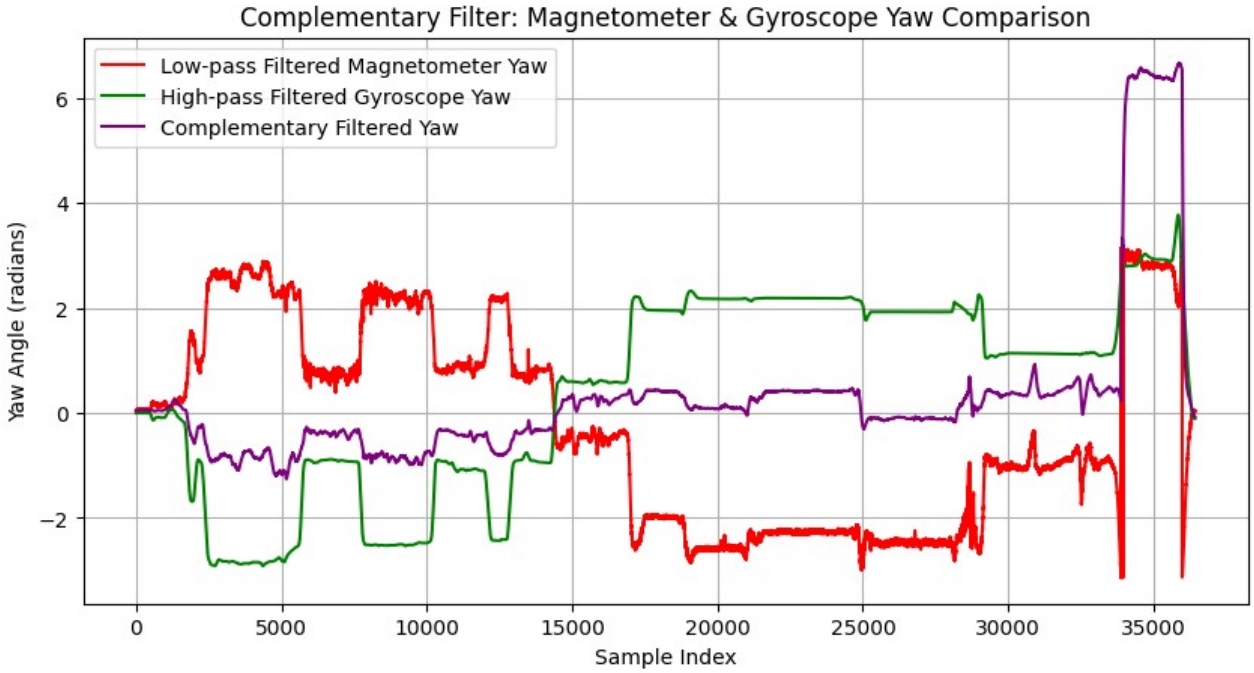

Fig – 5 Filter Comparison

Q3. Which estimate or estimates for yaw would you trust for navigation? Why?

The complementary filter output is the sum of low pass filter and high pass filter, smoothened out. It is evident from Fig 5 that complementary data fuses data from gyro and magnetometer providing us a reliable yaw angle. Also when the trajectory paths were estimated, complimentary filtered Yaw provided promising results.

Q4. What adjustments did you make to the forward velocity estimate and why?

The IMU velocity was obtained by integrating the forward acceleration which can be gathered from the IMU's accelerometer sensor. Initially there was a negative biasof the forward acceleration X of the IMU which you can see in Fig 6.

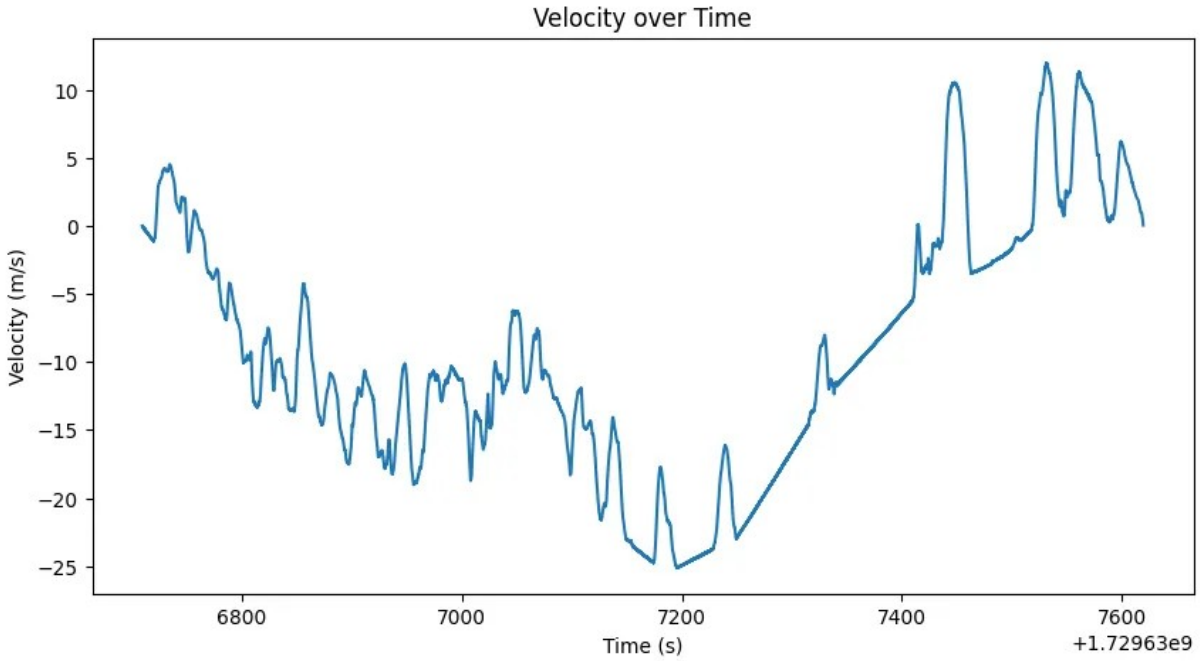

Fig – 6 Estimated IMU Velocity with Negative Bias

First we referenced the IMU timestamps to (0,0). Then we calculated the offset present in the accelerometer data.

Accelerometer offset = mean(Forward acceleration)

Corrected Acceleration = Forward Acceleration – Acceleration offset

$\int ¿ Corrected\ Acceleration\ .d(Timestamps\ for\ driving) > ¿$ = Forward Velocity IMU

For GPS velocity, we deployed the following method:

E_i = UTM_Easting with index i

t_i = Corresponding time for the above distance.

I calculated the unique GPS time instances ti_{+1} and its corresponding easting values E_{i+1}

GPS_Velocity = $\frac{E(i+1)-E(i)}{t(i+1)-t(i)}$

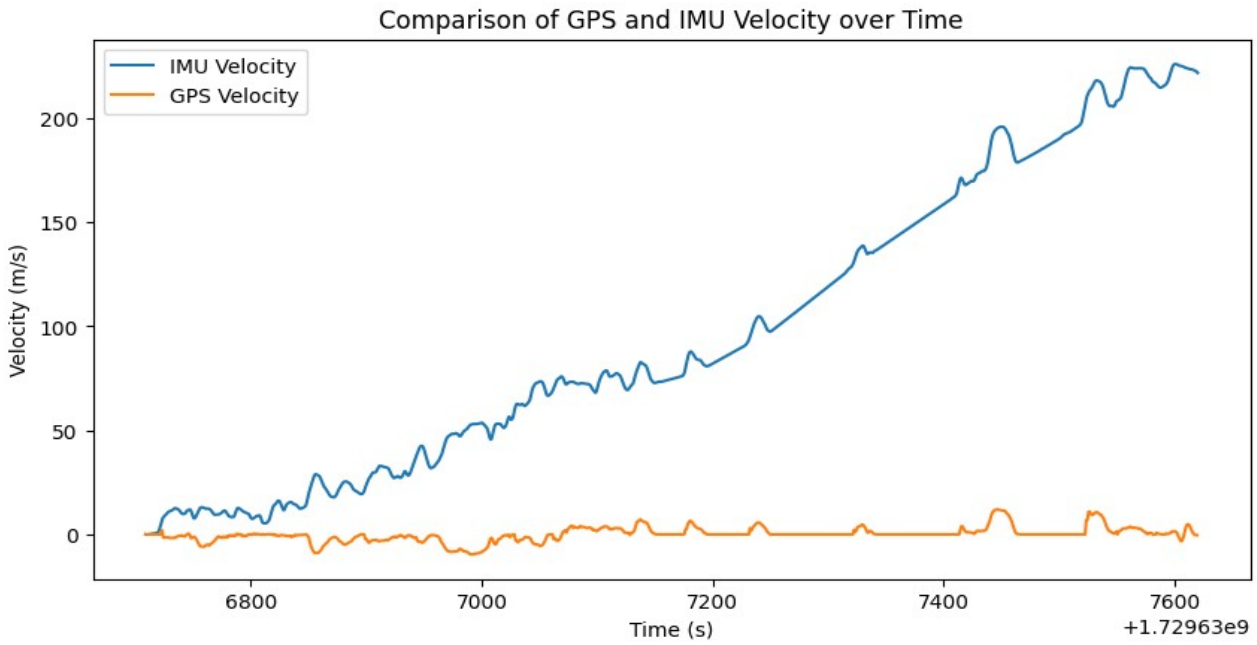

Fig – 7 Velocity Comparison of GPS and IMU

After calculating and correcting the required parameters, the velocities of GPS and IMU was plotted which is shown in Fig 7.

The positive bias present was rectified using a high pass filter with a cutoff frequency of 0.1 Hz and removing the negative values.

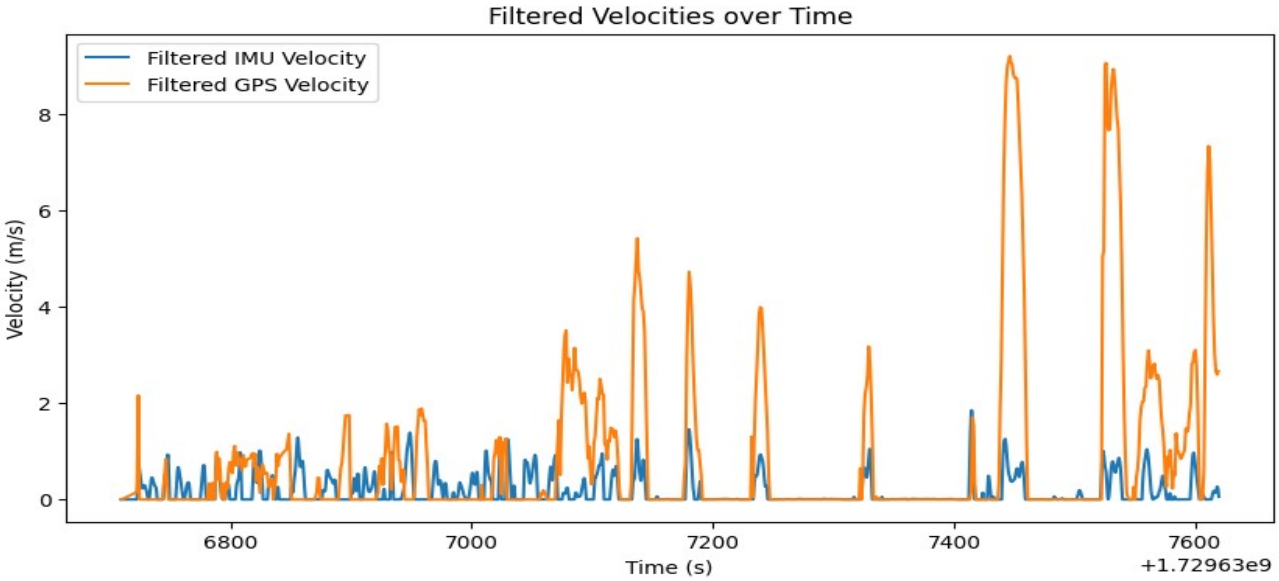

Fig – 8 Filtered Velocities

Q5. What discrepancies are present in the velocity estimate between acceleration and GPS? Why?

The discrepancies that were observed in the velocity estimation which we obtained by integrating acceleration is the presence of negative bias which was not observed in the GPS velocity. This maybe because of the fact that GPS points signifies zero acceleration during traffic hours while IMU sensors will show some error due to the presence of noise.

Q6. Compute $\omega\dot{X}$ and compare it to $\ddot{y}_{obs}$. How well do they agree? If there is a difference, what is it due to?

The $\omega\dot{X}$ and $\ddot{y}_{obs}$ are calculated using the below formula:

$$\ddot{x}_{obs} = \ddot{X} - \omega\dot{Y} - \omega^2 x_c$$
$$\ddot{y}_{obs} = \ddot{Y} + \omega\dot{X} + \dot{\omega}\, x_c$$

As per the document, the notation taken for the position of center of mass of the vehicle and the rotation rate about the center of mass CM (0,0, ω), The position of the inertial sensor in space is denoted by (x,y,0) and it's position in the vehicle frame is (X_c,0,0).

By applying these formula and substituting the necessary values, Y was calculated. We used a low pass filter to cancel out noise.

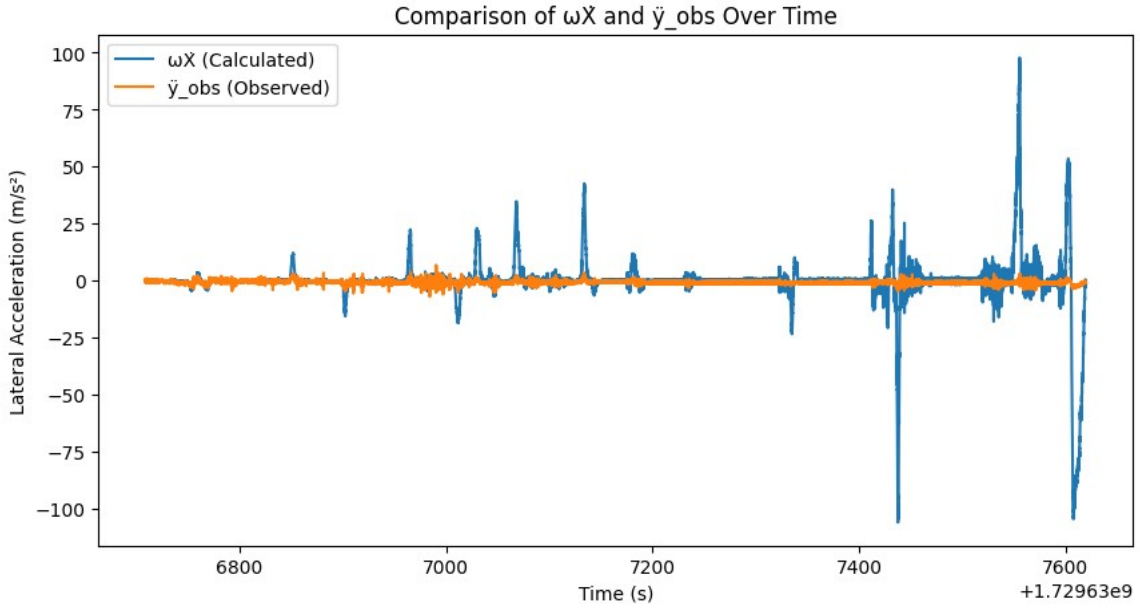

Fig – 9 Dead Reckoning using IMU and GPS

Q7. Estimate the trajectory of the vehicle (xe,xn) from the inertial data and compare it with GPS. (adjust heading so that the first straight line from both are oriented in the same direction). Report any scaling factor used for comparing the tracks.

For the estimation of trajectory, the IMU's forward velocity was decomposed further to easting and northing components using the aligned yaw.

Ve = Velocity*cos(complimentary filtered yaw)

Vn = Velocity*sin(complimentary filtered yaw)

The angle offset is the difference between the direction of the GPS trajectory and the initial yaw from the IMU.

The rotation applies this angle to the IMU data, transforming the IMU's coordinates into a frame that is aligned with the GPS coordinates. This is done using a 2D rotation matrix.

Finally, the displacments were found by integrating both the velocity components with respect to time.

These GPS and IMU paths were converted into relative coordinates to better visualize the trajectory's shape and orientation

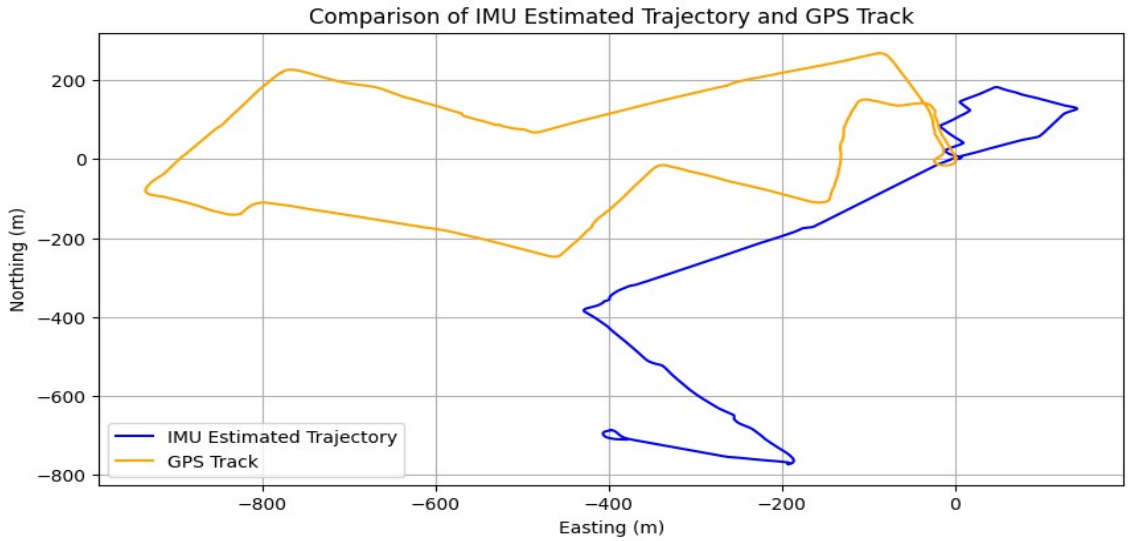

Fig – 10 Vehicle Trajectory

Q8. Given the specifications of VectorNav, how long would you expect that it is able to navigate without a position fix? For what period did your GPS and IMU estimates of position match closely (within 2m)? Did the stated performance for dead reckoning match actual measurments? Why or why not?

Given the specifications of VectorNav, I don't see it performing well because it doesn't operate with a feedback mechanism to track the errors. The IMU and GPS estimates closely match but only for few seconds. If position fix is applied we would have gotten identical results as a general trend is being followed throughout the trajectory.

Q9. Estimate Xc and explain your calculations (bonus up to 100%)

The Xc value is estimated to be 0.28 meters